#include "lcd.h"

// lets start with that enable pulse

static void lcd\_enable\_pulse(void)

{

LCD\_EN\_PORT |= (1 << LCD\_EN\_PIN); // enable pin =1

\_delay\_us(1); // these delay values come from datasheet

LCD\_EN\_PORT &= ~(1 << LCD\_EN\_PIN); // enable pin =0

\_delay\_us(50); // allow LCD to latch

}

// this function sends a nibble to the data pins (b3..b0 -> D7..D4)

static void lcd\_send\_nibble(uint8\_t nib)

{

// Clear PD2..PD5 by our mapping

LCD\_D\_PORT &= ~((1<<LCD\_D7\_PIN)|(1<<LCD\_D6\_PIN)|(1<<LCD\_D5\_PIN)|(1<<LCD\_D4\_PIN)); // Making all data pins 0 initially

// now set the pins according to the data/command

if (nib & 0x8) LCD\_D\_PORT |= (1<<LCD\_D7\_PIN);

if (nib & 0x4) LCD\_D\_PORT |= (1<<LCD\_D6\_PIN);

if (nib & 0x2) LCD\_D\_PORT |= (1<<LCD\_D5\_PIN);

if (nib & 0x1) LCD\_D\_PORT |= (1<<LCD\_D4\_PIN);

lcd\_enable\_pulse();

}

// this function allows us to send a byte

static void lcd\_send\_byte(uint8\_t value, uint8\_t rs) // here we receive data and rs value

{

if (rs==1)

{

LCD\_RS\_PORT |= (1<<LCD\_RS\_PIN); // IT’S A Data. SO RS pin =1

}

else

{

LCD\_RS\_PORT &= ~(1<<LCD\_RS\_PIN); // it’s a Command. So RS pin0

}

lcd\_send\_nibble(value >> 4); // sending upper nibble

lcd\_send\_nibble(value & 0x0F); // sending lower nibble

}

// this function sends commands (instructions)

void lcd\_cmd(uint8\_t c)

{

lcd\_send\_byte(c, 0);

// Clear (0x01) and Return Home (0x02) need longer delays in write-only mode

if (c == 0x01 || c == 0x02)

\_delay\_ms(2);

}

// This function allows us to send data

void lcd\_data(uint8\_t d)

{

lcd\_send\_byte(d, 1);

}

// this function allows us to set the cursor

void lcd\_set\_cursor(uint8\_t row, uint8\_t col) // receives row and column value 2X16 LCD

{

uint8\_t addr = (row ? 0x40 : 0x00) + (col & 0x0F); // this is a ternary operator.

lcd\_cmd(0x80 | addr); // send (80 + addr) as a command.

}

//This function allows us to print a string

void lcd\_print(const char \*s)

{

while (\*s)

{

lcd\_data((uint8\_t)\*s++);

}

}

// this function allows us to clear the lcd

void lcd\_clear(void)

{

lcd\_cmd(0x01);

}

// this function allows us to return the cursor to home

void lcd\_home(void)

{

lcd\_cmd(0x02);

}

// this function initialises the lcd. (initialisation sequence from datasheet)

void lcd\_init(void)

{

// Make control and data pins outputs

LCD\_RS\_DDR |= (1<<LCD\_RS\_PIN);

LCD\_EN\_DDR |= (1<<LCD\_EN\_PIN);

LCD\_D\_DDR |= (1<<LCD\_D7\_PIN)|(1<<LCD\_D6\_PIN)|(1<<LCD\_D5\_PIN)|(1<<LCD\_D4\_PIN);

\_delay\_ms(20); // power-up wait

LCD\_RS\_PORT &= ~(1<<LCD\_RS\_PIN); // RS=0

// 8-bit wake-up sequence (sent as high nibbles). why nibbles?

LCD\_EN\_PORT &= ~(1 << LCD\_EN\_PIN); // make EN =0 .

lcd\_send\_nibble(0x03);

\_delay\_ms(5);

lcd\_send\_nibble(0x03);

\_delay\_us(150);

lcd\_send\_nibble(0x03);

\_delay\_us(150);

//Switch to 4-bit

lcd\_send\_nibble(0x02);

\_delay\_us(150);

// note that from here onwards its cmd !

//Function set: 4-bit, 2 lines, 5x8 font

lcd\_cmd(0x28);

//Display off

lcd\_cmd(0x08);

//Clear

lcd\_cmd(0x01);

//Entry mode: increment, no shift

lcd\_cmd(0x06);

// Display on, cursor off, blink off

lcd\_cmd(0x0C);

}

// function to print integers. (we use recursion)

void lcd\_print\_uint16(uint16\_t v)

{

if (v >= 10)

{

lcd\_print\_uint16(v / 10); // print higher digits first

}

lcd\_data('0' + (v % 10)); // then print the last digit

}